

```

1
2 #include <Wire.h>    //
3
4 // Pas besoin d'autre bibliothèque spécifique au BNO055. Il suffit d'écrire les
valeurs spécifiées dans la notice aux adresses des registres de commande.
5
6 // Adresses des registres de commande et de lecture: Toutes ces adresses ne sont pas
utiles dans ce programme. Peuvent servir pour d'autres applications)
7 // Celles utiles sont visibles dans le sous-programme "void config_BNO055_Euler()"
8
9 // Definition des adresses des registres de commande
10 #define PAGE_SWAP      0x07 // page 0 ou 1: pour choisir la page mémoire du BNO055
[tableau page 58(depuis la page 0) et page 59 (depuis la page 1)]
11 #define ACC_CONF      0x08 // pag 1: pour configurer l'accéléromètre /
ACC_power_mode, ACC_BW, ACC_range -valeurs au § 3-5-2 page 29
12 #define GYR_CONF_0    0x0A // page 1: pour configurer le gyromètre /
GYR_bandwidth, GYR_range, GYR_power_mode - valeurs au § 3.5.3 page 30
13 #define GYR_CONF_1    0x0B // page 1: pour configurer le gyromètre /
GYR_bandwidth, GYR_range, GYR_power_mode - valeurs au § 3.5.3 page 30
14 #define MAG_CONF      0x09 // page 1: pour configurer le magnétomètre /
GYR_bandwidth, GYR_range, GYR_power_mode - valeurs au § 3.5.4 page 31
15 #define TEMP_SOURCE   0x40 // page 0: pour configurer la source de température
(Accéléro ou Gyro) page 39
16 #define UNIT_SEL      0x3B // page 0: Selection des unités pages 56, 75 ****
utilisée dans ce programme*****
17 #define PWR_MODE      0x3E // page 0: power mode selection page 56, 76, 20
****utilisée dans ce programme*****
18
19 // Definition des adresses des registres de lecture
20 #define HEADING        0x1A // page 0: pour lecture Euler heading LSB (les
adresses pour Roll et Pitch suivent) *** utilisée dans ce programme***
21 #define OMEGA_X        0x14 // page 0: pour lecture GYR_DATA_X LSB (les
adresses pour GYR_DATA_Y et GYR_DATA_Z suivent) - non utilisée dans ce programme
22
23 // #define MODE_REG      0x3D // initial
24 #define OPR_MODE       0x3D // page 0 / pages 56, 76, 22
25 #define FUSION_NDOF    0x0C // Fusion mode with NDOF (Nine Degrees Of Freedom) page 22
26 #define GYRO_ONLY      0x03 // Gyro only mode page 22
27 #define FUSION_IMU     0x08 // Fusion mode with IMU (Gyro + accelero) page 22
28
29 // Definition adresse I2C du BNO005
30 // #define ADDRESS      0x29 // adresse initiale de l'exemple
31 #define ADDRESS        0x28 // adresse fournie par GoTronic
32
33 word head = 0;
34 byte data[] = {0,0,0,0,0,0};
35 word Azim[100];
36 int Site[100];
37 int site;
38 int i=0; // compteur en mode enregistrement
39 int mode = 0; // mode = 0: mode chasse mode =1: mode enregistrement =>
toujours initialisé en mode chasse
40 int verif_serial = 1 ; // verif_serial = 1 si on veut un affichage sur serial
verif_serial = 0 si on n'en veut pas
41 int verif_enreg = 0; // Si verif_serila = 1, vérification du vecteur qui vient
d'être enregistré verif_enreg =0 ou verif_enreg = 1
42 int recorded = 0; // Y a-t-il eu enregistrement ? non = 0 oui = 1
43 int j0; // valeur particulière de j
44 int j1;
45 int azi_1; // borne gauche de l'azimut
46 int azi_2; // borne droite de l'azimut
47
48 // *****
49 void setup()
50 {
51     Wire.begin();
52     Serial.begin(19200);
53
54     delay(20);
55     config_BNO055_Euler();
56     delay(1000);
57     Serial.println();
58     Serial.println("Bonjour");

```

```

59     Beep(); Beep2();Beep(); Beep2();
60 }
61
62 // *****
63 void loop()
64 {
65     //           Mesure Angles d'Euler
66     delay(200); // période d'échantillonnage d'environ 200 ms pendant
                  l'enregistrement.
67
68     Wire.beginTransaction(ADDRESS); // Starts communication with cmpr03
69     Wire.write(HEADING); // Sends the register we wish to read
70     Wire.endTransmission();
71
72     Wire.requestFrom(ADDRESS, 6);
73
74     while(Wire.available() < 6);
75
76     data[0] = Wire.read(); // Heading LSB
77     data[1] = Wire.read(); // Heading MSB
78     data[2] = Wire.read(); // Roll LSB
79     data[3] = Wire.read(); // Roll MSB
80     data[4] = Wire.read(); // Pitch LSB
81     data[5] = Wire.read(); // Pitch MSB
82
83     short head = (short)((short)data[1]<<8 | (short)data[0]);
84     short roll = (short)((short)data[3]<<8 | (short)data[2]);
85     short pitch = (short)((short)data[5]<<8 | (short)data[4]);
86
87     short mod = (((head % 16)*10)/16);
88
89     if (verif_serial == 1) {
90         Serial.print("Direction ");
91         Serial.print(head/16);
92         Serial.print(".");
93         Serial.print(mod);
94
95         Serial.print("    Roll ");
96         Serial.print(roll/16);
97
98         Serial.print("    Pitch ");
99         Serial.print(pitch/16);}
100
101     // *****
102     if (pitch/16 >45) // entrée en mode enregistrement si l'élévation devient
                      supérieure à 45 degrés (valeur d'élévation hors plage utile)
103     {
104         if (mode == 0) {mode = 1; pitch = 0; recorded = 0;
105             for (int j =0; j <=99 ;j++) {Azim[j]= 0; Site[j]= 0;} // remise à zéro des
                      vecteurs Azim et Site
106             Beep();delay(50);Beep();delay(50);Beep(); i=0;}
107             delay(3000);Beep();
108         }
109
110         if (mode == 1) { Azim[i]= head/16; Site[i] = pitch/16; i=i+1; verif_enreg = 1;}
111         // enregistrement des points à chaque tour de boucle Loop
112
113     if (i == 75) // Sortie du mode enregistrement lorsque 76 mesures ont
                  été recueillies
114     {
115         if (mode == 1) {mode = 0; pitch = 0; i = 0; Beep();}
116         azi_1 = Azim[i];
117         for (int j = 1; j <= 99; j=j+1) { if (Azim[j] == 0) {azi_2 = Azim[j-2]; break;} }
118         recorded = 1;
119         delay(2000); goto fin_boucle;
120     }
121
122     if (verif_serial == 1){
123         Serial.print("    Mode "); Serial.print(mode); Serial.print("    azi_1 ");
124         Serial.print(azi_1); Serial.print("    azi_2 "); Serial.print(azi_2);
125
126         if ((mode == 0) && (verif_enreg == 1) ){ // on vérifie que Azim et Site

```

```

126     sont bien remplis et que après les données enregistrées tout est à zéro
127     for (int j =0; j<=99 ;j++) {Serial.print(Azim[j]); Serial.print("    ");
128     Serial.println(Site[j]);}
129     verif_enreg = 0; }
130
131     if (mode == 0)
132     {
133         for (int j = 1; j <= 99; j=j+1) {if (Azim[j] >= head/16) {j1 = j; site =
134         Site[j1]; break;}} // Pour une interpolation plus précise, tenir compte que
135         Azim[j] et Azim[j+1] sont parfois égaux..
136         if (recorded == 1) { if ( (head/16 <= azi_1) | (head/16 >= azi_2) )
137         {Beep();delay(50);Beep();} }
138         if (recorded == 1) { if ( pitch/16 >= site) {Beep(); delay(50); Beep(); } }
139
140         if (verif_serial ==1) {Serial.print("    Site "); Serial.print(site);
141         Serial.print("    J "); Serial.println(j1); }
142
143     fin_boucle:
144     delay(1);
145     }
146
147     //*****
148     *****
149     void config_BNO055_Euler()
150     // Tout ce qui ne sert pas dans ce programme a été mis en commentaire
151     {
152     // Wire.beginTransaction(ADDRESS);
153     // Wire.write(PAGE_SWAP); // Page swap
154     // Wire.write(1); // To page 1
155     // Wire.endTransmission();
156     //
157     // Wire.beginTransaction(ADDRESS);
158     // Wire.write(ACC_CONF); // ACC Configuration
159     // Wire.write(0x08); // 0x08 acc conf| normal power, 31.25hz
160     bandwidth, 2G range
161     // Wire.endTransmission();
162     //
163     // Wire.beginTransaction(ADDRESS);
164     // Wire.write(GYR_CONF_0); // GYR Configuration
165     // Wire.write(0x23); // 0x32 250DPS scale, 23hz bandwidth
166     // Wire.endTransmission();
167     //
168     // Wire.beginTransaction(ADDRESS);
169     // Wire.write(GYR_CONF_1); // GYR Configuration
170     // Wire.write(0x00); // Normal power
171     // Wire.endTransmission();
172     //
173     // Wire.beginTransaction(ADDRESS);
174     // Wire.write(MAG_CONF); // MAG Configuration
175     // Wire.write(0x1B); // Normal power, High accuracy, 10Hz
176     output rate
177     // Wire.endTransmission();
178
179     Wire.beginTransaction(ADDRESS);
180     Wire.write(PAGE_SWAP); // Page swap
181     Wire.write(0); // To page 0
182     Wire.endTransmission();
183
184     // Wire.beginTransaction(ADDRESS);
185     // Wire.write(TEMP_SOURCE); // Temperature source
186     // Wire.write(0x01); // gyro
187     // Wire.endTransmission();
188
189     Wire.beginTransaction(ADDRESS);
190     Wire.write(UNIT_SEL); // Unit select
191     Wire.write(0x01); // Temp in Degrees C, Heading in Degrees,
192     Angular rate in DPS, Gravity in mg
193     Wire.endTransmission();
194
195     Wire.beginTransaction(ADDRESS);

```

```

189     Wire.write(PWR_MODE); // Power mode normal
190     Wire.write(0x00); // normal
191     Wire.endTransmission();
192
193     Wire.beginTransaction(ADDRESS);
194     Wire.write(OPR_MODE); // Operation mode (page 21 de la notice)
195     // Wire.write(FUSION_NDOF); // Fusion avec boussole Ce mode ne pouvait
    pas convenir à cause de la masse métallique du canon du fusil
196     Wire.write(FUSION_IMU); // Fusion sans boussole Purement
    inertielle, pas de boussole utilisée.
197     Wire.endTransmission();
198 }
199
200 // *****
201 void Beep()
202 {
203     pinMode(10,OUTPUT);
204     for (int i=0;i<100;i++)
205     {digitalWrite (10, LOW);
206     delay(1);
207     digitalWrite (10, HIGH);
208     delay(1);}
209 }
210
211 void Beep2 ()
212 {
213     pinMode(10,OUTPUT);
214     for (int i=0;i<50;i++)
215     {digitalWrite (10, LOW);
216     delay(2);
217     digitalWrite (10, HIGH);
218     delay(2);}
219 }
220
221
222
223
224

```